운영체제

03분반

201402447 한원희

```
 puts("Please wait until server is ready...:");

 fd1 = open(MMAP1_NAME, O_RDWR|O_CREAT, 0664);
 fd2 = open(MMAP2_NAME, O_RDWR|O_CREAT, 0664);

 write(fd1, &zero, sizeof(int));
 write(fd2, &zero, sizeof(int));

 p_mmap = mmap(NULL, sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED,
;
 result = mmap(NULL, sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED,
;

 sem_unlink("client_sema");

 if((sem = sem_open("client_sema", O_CREAT|O_EXCL, 0644, 1)) ==
LED){
         perror("open");
         exit(1);
 }
```

server.c 먼저 확인하겠습니다. 서버를 준비하기 위해 공유메모리 설정과 semaphore 설정을 해줍니다. Semaphore가 열리지 않았을때를 위한 예외처리 또한 포함되어 있습니다. 맨 처음 공유메모리 두개, p_mmap과 result에는 0이 적혀져 있습니다.

```
sem_wait(sem);

sleep(1);

while(1){       // player1
        sleep(1);
        sem_getvalue(sem, &client1);
        if(client1 == 1){
                printf("Client %d hi with slot 0\n",*(p_mmap));
                client1 = *(p_mmap);
                break;
        }
}

sem_wait(sem);

while(1){       // player2
        sleep(1);
        sem_getvalue(sem, &client2);
        if(client2 == 1){
                printf("Client %d hi with slot 1\n",*(p_mmap));
                client2 = *(p_mmap);
                break;
        }
}
```

그 이후인 플레이어들을 기다리는 과정입니다. Semaphore의 를 wait 시켜줍니다. Client가 실행되면

```
if((sem = sem_open("client_sema", O_EXCL)) == SEM_FAILED){
        perror("sem_open failed.");
        puts("An error is accured, tell admin.");
        exit(1);
}

sem_post(sem);

pid = getpid();
*(p_mmap) = pid;

puts("waiting for server..\n");
puts("please wait for 3 seconds for synchronization");
sleep(3);
```

이런식으로 client.c에서 sem_post를 시켜줍니다. 그 이후 공유메모리

p_mmap에 자신의 pid를 저장시켜 줍니다. 다시 server.c로 돌아가면 sem_getvalue 함수로 현재 semaphore가 post되어 있는지 확인해줍니다. 되어있다면 공유 메모리 p_mmap에 있는 client의 pid를 출력해주고 두번째 플레이어를 기다리기 위해 sem_wait을 해줍니다. Client pid는 client1, client2 에 저장해줍니다.

```
for(int i = 1; ;i++){

        sem_wait(sem);

        if(*(result) != 0){
                printf("[%d]", *(result));
        }
        if(*(p_mmap) != 0){
                printf("%d\n", *(p_mmap));
        }

        if(i%10==3 | i%10==6 | i%10==9){
```

그 이후 server에서 받아온 값을 출력해주고, 검사를 하는 단계입니다. result에는 현재 그 client의 pid값이 들어가 있습니다.

```
if(i%10==3 | i%10==6 | i%10==9){
        if(*(p_mmap) != -1){

                if(client1 == *(result)){
                        printf("Player %d lose!\n", client1);
                        kill(client1,SIGUSR2);
                        kill(client2,SIGUSR1);
                        printf("Client %d out\n", client1);
                }
                else{
                        printf("Player %d lose!\n", client2);
                        kill(client1,SIGUSR1);
                        kill(client2,SIGUSR2);
                        printf("Client %d out\n", client2);
                }

                exit(1);
        }
}
```

받은 값을 검사해줍니다. 숫자에 3, 6, 9가 들어가 있다면 client의 pid와 lose를 출력해주고 kill 함수를 통해 signal을 보내줍니다.

```
else{
        if(*(p_mmap) != i){

                if(client1 == *(result)){
                        printf("Player %d lose!\n", client1);
                        kill(client1,SIGUSR2);
                        kill(client2,SIGUSR1);
                        printf("Client %d out\n", client1);
                }
                else{
                        printf("Player %d lose!\n", client2);
                        kill(client1,SIGUSR1);
                        kill(client2,SIGUSR2);
                        printf("Client %d out\n", client2);
                }
                exit(1);
        }
}
```

받은 값이 현재 for문의 숫자와 같지 않다면 kill을 통해 signal을 보내줍니다.

```
    client_pid = *(result);

    *(result) = *(p_mmap);
    *(p_mmap) = client_pid;

    sem_post(sem);

    sleep(1);
```

현재 입력받은 client의 pid를 client_pid에 넣어주고 result에 입력받은 값을 저장해줍니다. 그리고 client_pid는 p_mmap에 저장해주고 sem을 마칩니다. 이후에 signal이 보내질때까지 무한반복 합니다.

```
        p_mmap = mmap(NULL, sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED,
fd1, 0);
        result = mmap(NULL, sizeof(int), PROT_READ|PROT_WRITE, MAP_SHARED,
fd2, 0);

        if((sem = sem_open("client_sema", O_EXCL)) == SEM_FAILED){
                perror("sem_open failed.");
                puts("An error is accured, tell admin.");
                exit(1);
        }

        sem_post(sem);

        pid = getpid();
        *(p_mmap) = pid;
```

Client.c입니다. 공유메모리와 semaphore를 생성해줍니다. 예외처리로 server가 시작되지 않았다면 오류 메시지를 출력합니다.

```
os_201402447@os03:~/week7$ ./client
sem_open failed.: No such file or directory
An error is accured, tell admin.
```

오류메시지 입니다.

생성을 마치면 현재 server에서 sem_wait을 해주었으니 sem_post로 풀어줍니다. 그 이후 p_mmap에 자신의 pid를 저장합니다.

```
    for(int i = 1; ;i++){
            sleep(1);
            sem_wait(sem);
            act.sa_handler = end;
            |
            sigaction(SIGUSR1, &act, NULL);
            sigaction(SIGUSR2, &act, NULL);
            sigaction(SIGINT, &act, NULL);

            if(*(p_mmap) != 0){
                    printf("[%d]", *(p_mmap));
            }
            if(*(result) != 0){
                    printf("%d\n", *(result));
            }

            printf("input num : ");
            scanf("%d", p_mmap);
            printf("[%d]", pid);
            printf("%d\n", *(p_mmap));
            *(result) = pid;

            sem_post(sem);

            sleep(1);

    }
```

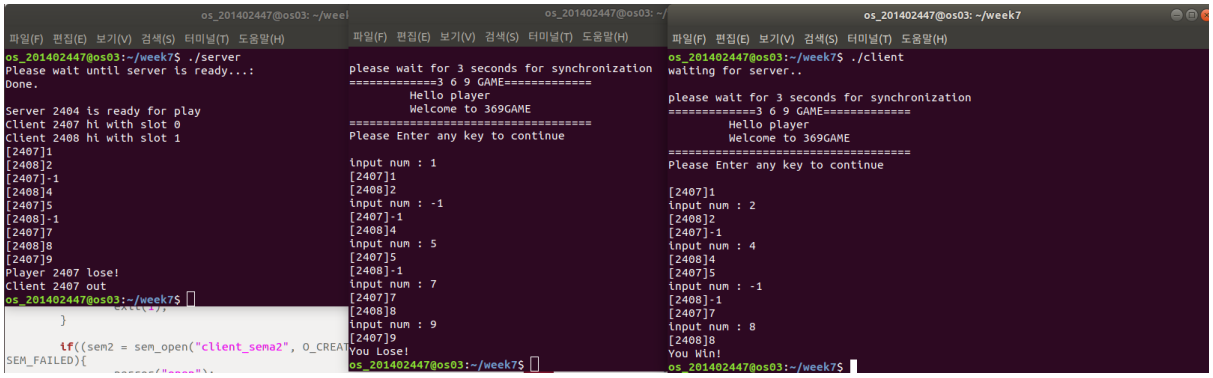
Act와 sigaction으로 미리 시그널이 왔을때를 대비해 줍니다.

```
void end(int sig){
        switch(sig){
                case SIGUSR1:
                        puts("You Win!");
                        exit(1);
                case SIGUSR2:
                        puts("You Lose!");
                        exit(1);
        }
}
```

시그널이 왔을 때 판단해주는 메소드입니다. server에서 이긴사람은 SIGUSR1을 보내고 진사람에겐 SIGUSR2를 보냅니다.

server에서 저장한 공유메모리를 통해 상대방의 pid와 값을 출력해줍니다. 그 이후 값을 입력받고 자신의 pid를 저장해주고 sem_post 해줍니다.

실행 화면


os_201402447@os03: ~/week7
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
os_201402447@os03:~/week7$ ./server
Please wait until server is ready...:
Done.
Server 2404 is ready for play
Client 2407 hi with slot 0
Client 2408 hi with slot 1
[2407]1
[2408]2
[2407]-1
[2408]4
[2407]5
[2408]-1
[2407]7
[2408]8
[2407]9
Player 2407 lose!
Client 2407 out
os_201402447@os03:~/week7$
please wait for 3 seconds for synchronization
============3 6 9 GAME============
Hello player
Welcome to 369GAME
==================================
Please Enter any key to continue
input num : 1
[2407]1
[2408]2
input num : -1
[2407]-1
[2408]4
input num : 5
[2407]5
[2408]-1
input num : 7
[2407]7
[2408]8
input num : 9
[2407]9
You Lose!
os_201402447@os03:~/week7$
os_201402447@os03: ~/week7
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
os_201402447@os03:~/week7$ ./client
waiting for server..
please wait for 3 seconds for synchronization
============3 6 9 GAME============
Hello player
Welcome to 369GAME
==================================
Please Enter any key to continue
[2407]1
input num : 2
[2408]2
[2407]-1
input num : 4
[2408]4
[2407]5
input num : -1
[2408]-1
[2407]7
input num : 8
[2408]8
You Win!
os_201402447@os03:~/week7$